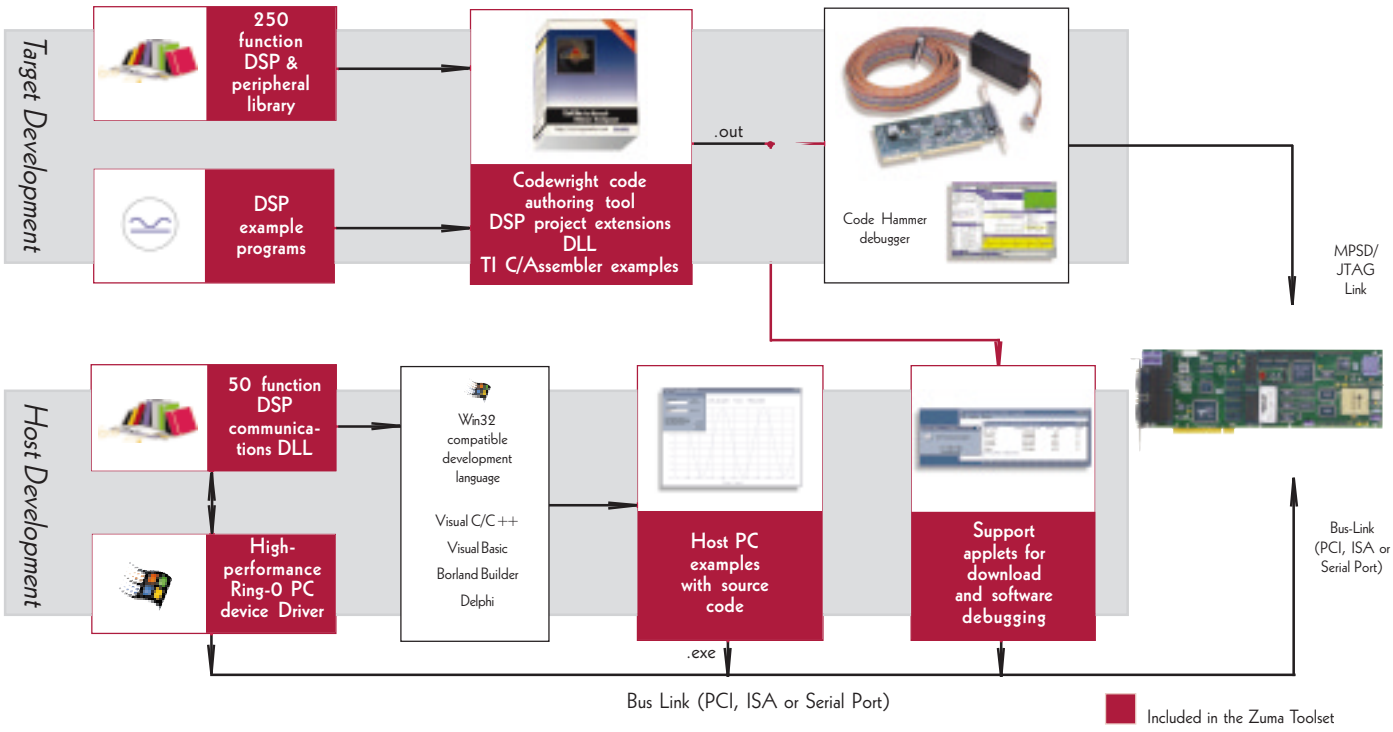
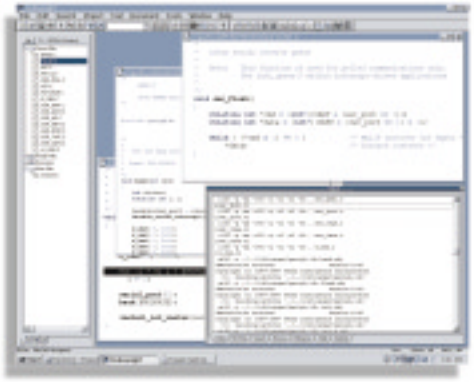
ZUMA TOOLSET

High-Performance Target Specific Libraries



The Zuma Toolset is a comprehensive, state-of-the-art collection of tools used to develop application programs for Innovative Integration's family of digital signal processor boards. The Toolset supports development of embedded DSP programs as well as 32-bit Windows 95/NT 4.0 host PC programs that interact with embedded DSP code in order to perform sophisticated, high-speed data acquisition, control and signal processing tasks.



The Codewright authoring tool is used as the cornerstone of the Zuma DSP code development environment. While Codewright emulates all popular editor keymaps and functions, it is much more than just another editor. Codewright supports advanced project management features including work spaces, multi-language syntax highlighting (including TI C and Assembler) multi-file text searches and replacements, automatic browse database creation and literally hundreds of other professional features needed in the development of complex embedded applications. Zuma further extends the capabilities of Codewright via a custom DLL (dynamic link library) that seamlessly creates and dispatches project make files for rapid regeneration of target executables when any project dependency is changed.

Features

- Codewright integrated code authoring environment including a world-class editor augmented by a custom DLL to provide makefile script generation and DSP software toolset interface
- Target DSP example programs in source form
- Sample applications showing host PC as well as target DSP coding techniques
- 250 function DSP & peripheral control library with full source code
- One full year of hot-line technical support

Windows Compatible

- High-performance Ring 0, 32-bit WindowsNT/95 device driver & dynamic link library with numerous PC example programs in source form
- Host support applets for automatic program download, terminal emulation, COFF file dumping and on-board flash programming

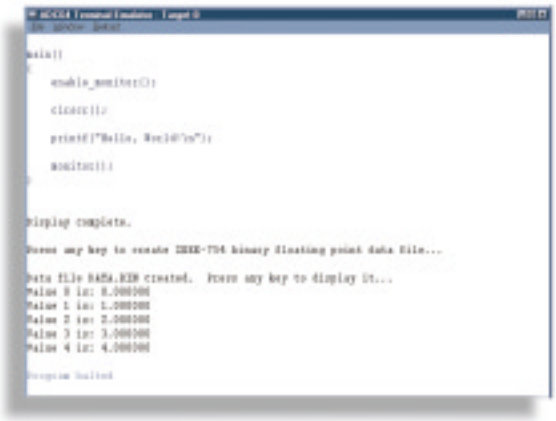
tel (818) 865-6150 • fax (818) 879-1770 • www.innovative-dsp.com

Innovative Integration

ZUMA TOOLSET

Zuma provides an exhaustive two-hundred and fifty function target-specific DSP and peripheral control library that greatly simplifies target DSP application development. Plus, the library routines are amply illustrated via dozens of simple example programs. Both the libraries and examples are provided in full source form. Using the peripheral libraries makes development of complex DSP applications a snap!

Category	*Available Routines*
Digital Signal Processing	*FIR filters, Forward, inverse real and complex FFTs, Windowing, vector operations.*
Math	*Matrix arithmetic, statistics calculation.*
Device Control	*High-performance real-time, A/D and D/A access. Maximum speed digital I/O access.*
Timers	*Programmable interval timers, counters, watchdog and real-time clock support.*
Communications	*Bus master transfers, mailbox I/O. Full ANSI C compatible standard I/O. Software-based monitor (Talker).*
Misc	*CPU register access and control. Memory sizing. CPU speed detection. FIFO access. DMA register programming. Board initialization.*

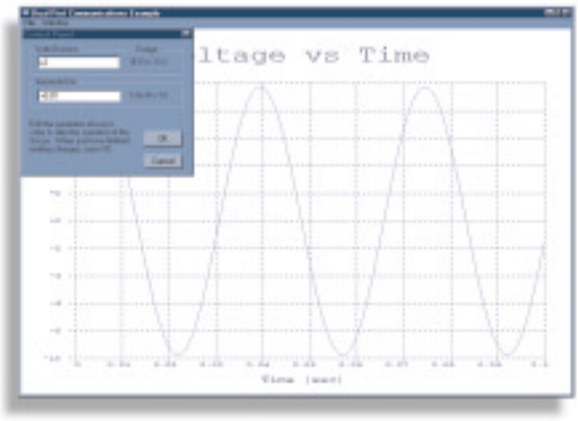


Once DSP code has been authored and successfully compiled, it is ready for debugging. Zuma provides numerous, sophisticated debugging tools to support both low- and high-level code penetration. Zuma is compatible with Code Hammer, Innovative's JTAG/MPSD hardware debugger, included in every Development Package.

Furthermore, Zuma supports advanced high-level ANSI-C standard I/O embedded directly into DSP code during the development cycle to accommodate intuitive console I/O, file I/O and real-time graphics. Additional utilities are provided to automate embedded file downloading during operating system start-up and burning embedded applications into FLASH on target boards featuring read-only memory.

Zuma support extends beyond target DSP development to include host PC code development as well. The Toolset includes a high-performance custom, Ring-O (kernel mode) device driver and a fifty-function DLL which supports optimal-performance communications with the embedded DSP board. The DLL/Driver provides all of the support functions needed to download code to the embedded DSP, control the card operation and implement bidirectional data communications at full bus bandwidth – up to 132 Mbytes/sec on PCI-based DSP boards!

A number of examples illustrating use of the DLL/Driver are supplied in Zuma. The example programs highlight everything ranging from host-to-target/target-to-host interrupts to common data passing techniques. One of the examples implements a real-time oscilloscope that plots the DSP's analog inputs in real-time. Using the supplied DLL functions, the target DSP control and data passing operations are reduced to just a dozen lines of Visual C code!



Zuma represents a real and substative breakthrough in DSP application code development, providing the first, complete, advanced tool suite for the generation of Innovative Integration DSP-based hardware-accelerated data acquisition, control and signal processing applications running under Windows 95 and NT 4.0.

Zuma Toolset . Host PC and Target DSP Functions

The following is a partial list of C-callable functions contained in the board-specific peripheral libraries within the Development Package for each base board. The first list covers target-callable functions, while the second covers host DLL functions. Each package includes numerous of example programs illustrating usage.

Target Procedure	Function
adc	Read A/D conversion results
adc_convert	Start conversion on A/D
baud	Set baud rate on current serial port
bold	Set console text bold attribute
calibrate_analog	Command A/D calibration and wait for completion
clreol	Clear console to end of line
clrscr	Clear console screen
cpu	Set CPU number and mailbox
cpu_number	Get CPU number
cpu_speed	Derive DSP clock speed
cursor	Enable/disable console cursor
dac	Write new data to the D/A's
dac_convert	Update D/A outputs
deinstall_int_vector	Remove vector from vector table
disable_interrupt	Disable specific interrupt
emit	Send a character to the terminal emulator
enable_analog	Initialize analog subsystem
enable_interrupt	Enable specific interrupt
fclose	Close a host disk file
fcreate	Open a host disk file for write
ffft_rl	Forward Fast Fourier Transform - Real
fir	Finite Impulse Response Filter
flush	Clear the current serial port's receive buffer
from_ieee	Convert from IEEE-754 Floating point format
fopen	Open a host disk file for read
fread	Read from host disk file via dual port RAM
fwrite	Write to host disk file via dual port RAM
get_abits	Retrieve current ABITS output values
get_attribute	Get current console text attribute type
get_DIE	Retrieve 320C4x DIE register
get_gain_a	Retrieve current A/D channel A gain setting
get_gain_b	Retrieve current A/D channel B gain setting
get_IE	Retrieve 320C3x IE register
get_IIE	Retrieve 320C4x IIE register
get_IF	Retrieve 320C3x IF register
get_IIF	Retrieve 320C4x IIF register
get_ST	Retrieve 320C3x/4x Status register
get_mailbox	Get access to dual port RAM mailbox
get_mux_a	Retrieve current A/D channel A multiplexer setting
get_mux_b	Retrieve current A/D channel B multiplexer setting
get_semaphore	Get hardware semaphore
get_ST	Retrieve 320C3x/4x ST register
getchar	ANSI get character from console
getint	Get integer from console
gets	ANSI gets from console
gotoxy	Set cursor position
ifft_rl	Inverse Fast Fourier Transform - Real
init_queue	Initialize the serial I/O receive queue
init_serial	Initialize the serial I/O system
install_interrupt_vector	Install vector into vector table
kbd_hit	Check for an available keystroke at the console

Target Procedure	Function
kbd_key	Get a key from the terminal emulator
monitor_mailbox	Set current dual port RAM mailbox
ms	Dwell milliseconds
normal	Set console text normal attribute
packb	Pack byte value into int
packh	Pack half word value into int
poll_ser_kbd_hit	Check for available keys on the current serial port
printf	ANSI printf to console
putchar	ANSI put character to console
putint	Send integer to console
puts	ANSI puts to console
random	Return positive random number less than ceiling
r_cmd	Read 8530 read register
read_mailbox	Read from current dual port RAM mailbox, with handshaking
release_mailbox	Relinquish access to dual port RAM mailbox
release_semaphore	Release hardware semaphore
scanf	ANSI scanf from console
ser_emit	Transmit a single character out the console
ser_getchar	Get a key from the serial port
ser_kbd_hit	Check for available keys on the current serial port
ser_key	Get a character from the current serial channel (polled)
ser_putchar	Transmit a single character out the current serial port
ser_type	Transmit a character string out the current serial port
serial_port	Set current serial port
set_abits	Set the ABITS output bits
set_attribute	Set current console text attribute type
set_DIE	Set 320C4x DIE register
set_gain_a	Set current A/D channel A gain
set_gain_b	Set current A/D channel B gain
set_IE	Set 320C3x IE register
set_IF	Set 320C3x IF register
set_IIE	Set 320C4x IIE register
set_IIF	Set 320C4x IIF register
set_mux_a	Set current A/D channel A multiplexer setting
set_mux_b	Set current A/D channel B multiplexer setting
set_PC	Set processor program counter
set_ST	Set processor status register
sscanf	ANSI sscanf from console
sprintf	ANSI sprintf
terminal_mailbox	Set current dual port RAM mailbox
timer	Set hardware timer frequency
timers	Initialize skewed timer channels
to_ieee	Convert to IEEE-754 Floating point format
type	Send a character string to the terminal emulator
uclock	Get system millisecond timer value
unpackb	Unpack byte values from int
unpackh	Unpack half word values from int
us	Dwell microseconds
w_cmd	Write 8530 write register
wherexy	Get cursor position
write_mailbox	Write to current dual port RAM mailbox, with handshaking

Host Procedure	Function
ack	Acknowledges a target board service request
acked	Checks for acknowledge from target
disable_dpram	Disables dual port RAM on the ISA bus
emit	Transmits a terminal emulation character to the target board
enable_dpram	Enables dual port ram on the ISA bus
dp_fetch	Read number from dual port RAM
fetch	Read number from Target address space
get_mailbox	Gets ownership of the semaphore for the current mailbox
get_semaphore	Gets ownership of a semaphore flag
kee	Receives a terminal emulation character from the target board
monitor_mailbox	Use the monitor mailbox
target_reset	Places the target processor in reset
target_run	Takes the target processor out of reset
read_mailbox	Read from current mailbox, with handshaking
release_mailbox	Releases ownership of the semaphore for the current mailbox
release_semaphore	Releases ownership of a semaphore flag
run_free	Releases the target board from JTAG halt mode
check_outbox	Check for empty outgoing mailbox
clear_mailboxes	Clear communications mailboxes
get_semaphore	Obtain ownership of target hardware semaphore
host_interrupt_deinstall	Remove target to source interrupt handler
host_interrupt_disable	Disable target to host interrupts
host_interrupt_enable	Begin processing target to host interrupts
host_interrupt_install	Install target to host interrupt processing function
iicoffld	Download program to Target

Host Procedure	Function
from_ieee	Convert from host to TI FP format
own_semaphore	Check ownership of target hardware semaphore
read_mailbox	Reads data from an incoming mailbox
read_mb_terminate	Read mailbox with success/failure
release_semaphore	Relinquishes ownership of target hardware semaphore
request_semaphore	Request ownership of target hardware semaphore
start_app	Launch a DSP application
start_talker	Starts Talker program execution on the target
talker_download	Downloads a data section to target memory
talker_fetch	Fetch value from target memory
talker_store	Store value to target memory
target_cardinfo	Return address of target CARDINFO structure
target_check	Verifies presence of target in system
target_close	Close the target
target_control	Update bit in specified target control register
target_inport	Fetch value from specified target host I/O register
target_interrupt	Triggers a host-to-target processor interrupt
target_open	Open a DSP target board VxD for communication
target_opreg_inport	Fetch valve from specified operating space register
target_opreg_outport	Store valve from specified operating space register
target_outport	Store value to specified target host I/O register
target_reset	Place target processor in reset and clear communication mailboxes
target_run	Releases target from reset
to_ieee	Convert from TI to host FP format
write_mailbox	Write to target mailbox
write_mb_terminate	Write mailbox with success/failure

tel (818) 865-6150 • fax (818) 879-1770 • www.innovative-dsp.com